

Bücher mit Code online publizieren

Gert-Ludwig Ingold

 <https://github.com/gertingold/lit2024>

```
meine_liste[1] = 2  
print(meine_liste)
```

```
[1, 2, 3]
```

Neben der Möglichkeit mit einer beliebigen Liste `meine_liste` in Python möglichst mehrere Listenelemente auf einmal zu adressieren, über kann es sich entweder um eine gegebene Anzahl direkt aufeinanderfolgender Listenelemente oder mehrere Listenelemente mit einem festen Abstand handeln. In der folgenden Liste von Primzahlen werden alle Elemente ab Index 1, also ab dem zweiten Element, bis ausschließlich dem Index 5

```
primzahlen = [2, 3, 5, 7, 11, 13, 17, 19, 23]  
print(primzahlen[1:5])
```

```
[3, 5, 7, 11]
```

Gerade zu Beginn ist es oft ungewohnt, dass das Element zum zweiten Index nicht Teil der Auswahl ist, auch wenn wir dies von der `range()`-Funktion schon gewohnt sind. Man kann sich dieses Verhalten dadurch veranschaulichen, dass man sich den Index nicht als Nummerierung eines Elements vorstellt, sondern als Markierung »zwischen« den Listeneinträgen, wie es in [Abb. 6.1](#) gezeigt ist.

Anwendungsszenarien

Mein Anwendungsszenario

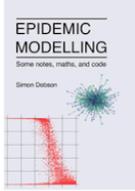
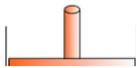
- Manuskript zu einem Programmierkurs frei zugänglich machen
- Präsentation auf einer responsiven Webseite
- Code wird zum Aufbau der Inhalte der Webseite ausgeführt und die Resultate werden integriert

Allgemeinere Szenarien

- allgemeine Lehrmaterialien mit Code zum Erzeugen von Abbildungen oder ganz ohne Code
- wissenschaftliche Arbeiten, z.B. mit Datenanalyse

Gallery of Jupyter Books

This is a gallery of Jupyter Books built from across the community. If you'd like to add your book to this list, simply [add an entry to this gallery.yml file](#) and open a Pull Request to add it.

<p>Epidemic modelling: Some notes, maths, and code</p>  <p>website repo Stars 19</p>	<p>Topology in condensed matter: tying quantum knots</p>  <p>website repo Stars 265</p>	<p>Quantitative Economics with Python</p>  <p>website repo Stars 75</p>
<p>Continuous Time Markov Chains</p>  <p>website repo Stars 7</p>	<p>Python Packages</p>  <p>website repo Stars 260</p>	<p>Machine Learning, Statistics, and Data Mining for Heliophysics</p>  <p>website repo Stars 66</p>
<p>Computational Thermodynamics</p> 	<p>Deep Learning for Particle Physicists</p> 	<p>Python For Data Science</p> 

Lecture notes »Tools for Scientific Computing«

- **1. Introduction**
- **2. Version Control with Git**
 - 2.1. Why version control?
 - 2.2. Centralized and distributed version control systems
 - 2.3. Getting help
 - 2.4. Setting up a local repository
 - 2.5. Basic workflow
 - 2.6. Working with branches
 - 2.7. Collaborative code development with GitLab
 - 2.8. Sundry topics
 - 2.8.1. Stashing
 - 2.8.2. Tagging
 - 2.8.3. Detached head state
 - 2.8.4. Manipulating history
- **3. Testing of code**
 - 3.1. Why testing?
 - 3.2. Doctests
 - 3.3. Testing with pytest
- **4. Scientific computing with NumPy and SciPy**
 - 4.1. Python scientific ecosystem
 - 4.2. NumPy
 - 4.2.1. Python lists and matrices
 - 4.2.2. NumPy arrays
 - 4.2.3. Creating arrays
 - 4.2.4. Indexing arrays
 - 4.2.5. Broadcasting
 - 4.2.6. Universal functions
 - 4.2.7. Linear algebra
 - 4.3. SciPy
- **5. Run-time analysis**

Sphinx

www.sphinx-doc.org

- in erster Linie zur Erzeugung von Dokumentation
- auch schöne PDF-Ausgabe
- Markup mit reStructuredText
- zahlreiche Erweiterungsmöglichkeiten
- Sprache einstellbar

- standardmäßig keine Codeausführung mit Einbettung des Ergebnisses
- Webseiten nicht responsiv

Beispiel links: gertingold.github.io/tools4scicomp



Aspekte werden wir später auf die von NumPy zur Verfügung gestellten Arrays übertragen können.

Listen sind uns beispielsweise bereits in [Kapitel 4.1](#) begegnet, wo wir die `range()`-Funktion verwendet hatten, um einen Schleifenzähler mit Werten zu versorgen. Dabei werden die benötigten Werte nur bei Bedarf erzeugt. Um alle Werte auf einmal zu sehen, hatten wir die `list()`-Funktion verwendet und dabei eine Liste erzeugt.

```
meine_liste = list(range(20))
print(meine_liste)
print(type(meine_liste))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
<class 'list'>
```

Mit der zweiten Ausgabezeile wird hier nachgewiesen, dass der Datentyp des Objekts `meine_liste` tatsächlich eine Liste ist.

Wenn man die Länge einer Liste nicht kennt, kann man diese mit Hilfe der `len()`-Funktion bestimmen.

```
liste1 = list(range(1, 17, 3))
print(f'Länge der ersten Liste: {len(liste1)} Elemente')
liste2 = ['Stein', 'Papier', 'Schere']
print(f'Länge der zweiten Liste: {len(liste2)} Elemente')
```

```
Länge der ersten Liste: 6 Elemente
Länge der zweiten Liste: 3 Elemente
```

Jupyter Book

jupyterbook.org | executablebooks.org

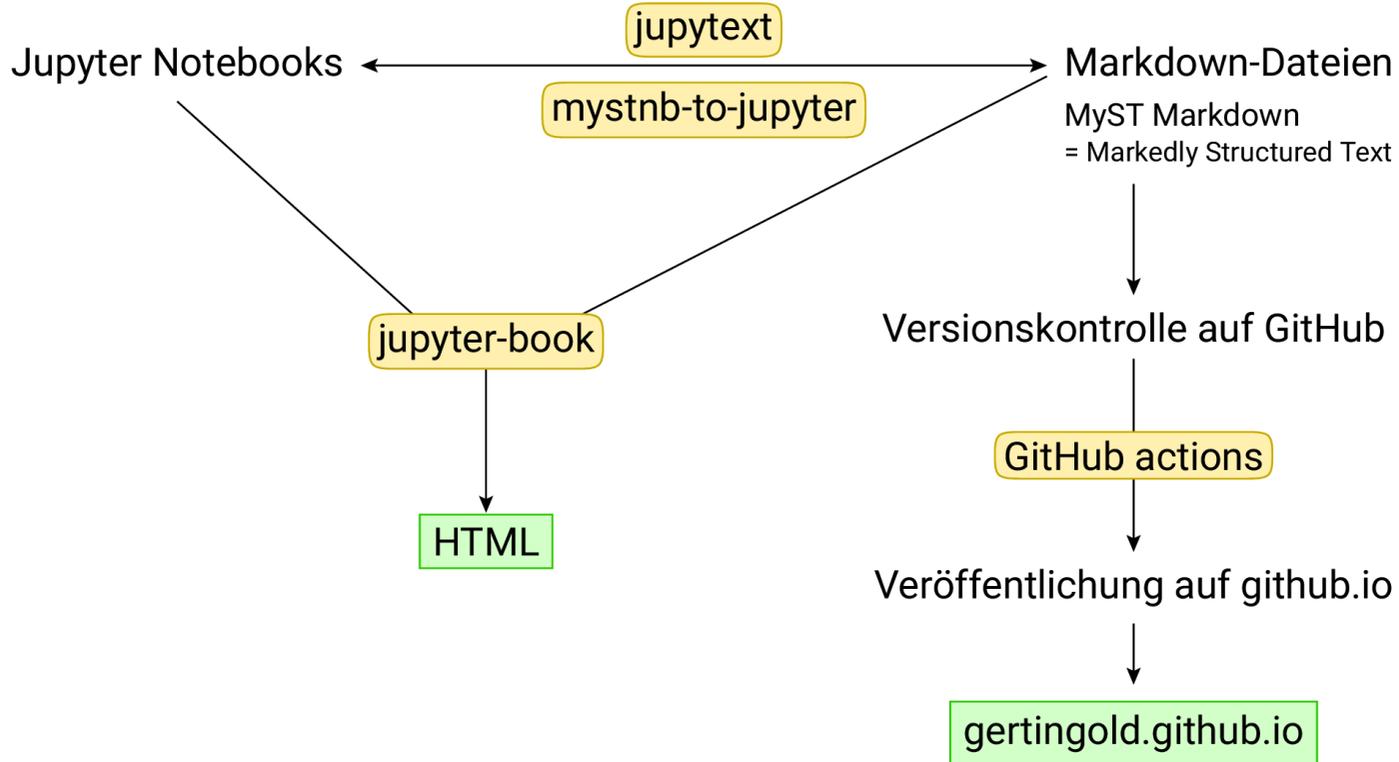
- basiert unter anderem auf Sphinx
- Sprache über die Sphinx-Konfiguration festlegbar

Listen sind uns beispielsweise bereits in `{numref}`forloop`` begegnet, wo wir die

- Ergebnis der Ausführung von Code lässt sich einbetten
- responsive Webseite

Beispiel links: gertingold.github.io/epriprog

Ein Gesamtüberblick



Harmonischer Oszillator

Zeitunabhängige Schrödingergleichung

$$-\frac{\hbar^2}{2m} \frac{d^2}{dx^2} \psi(x) + \frac{m}{2} \omega^2 x^2 \psi(x) = E \psi(x)$$

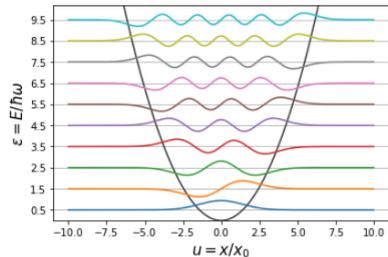
```
[1]: from math import factorial, pi, sqrt
```

```
import numpy as np
import numpy.linalg as LA
from scipy.special import eval_hermite
import matplotlib.pyplot as plt

import ipywidgets as widgets
from ipywidgets import interact
```

```
[2]: def ho_numerical(n, delta):
    u = delta*np.arange(-n, n+1)
    hamilton = -(np.eye(2*n+1, k=1) - 2*np.eye(2*n+1) + np.eye(2*n+1, k=-1))/delta**2
    hamilton = hamilton + np.diag(u**2/4)
    eigvals, eigvecs = LA.eigh(hamilton)
    return u, eigvals, eigvecs
```

```
n = 1000
delta = 0.01
u, eigvals, eigvecs = ho_numerical(n, delta)
plt.plot(u, u**2/4, '#3f3f3f')
plt.ylim(0, 10.2)
for n in range(10):
    psi = 7*eigvecs[:, n]/LA.norm(eigvecs[:, n])
    plt.plot(u, eigvals[n]+psi)
plt.xlabel('$u = x/x_0$', fontsize=15)
plt.ylabel(r'$\epsilon = E/\hbar\omega$', fontsize=15)
plt.yticks(np.arange(0.5, 10, 1))
plt.grid(axis='y')
```



Jupyter-Notebooks

jupyter.org

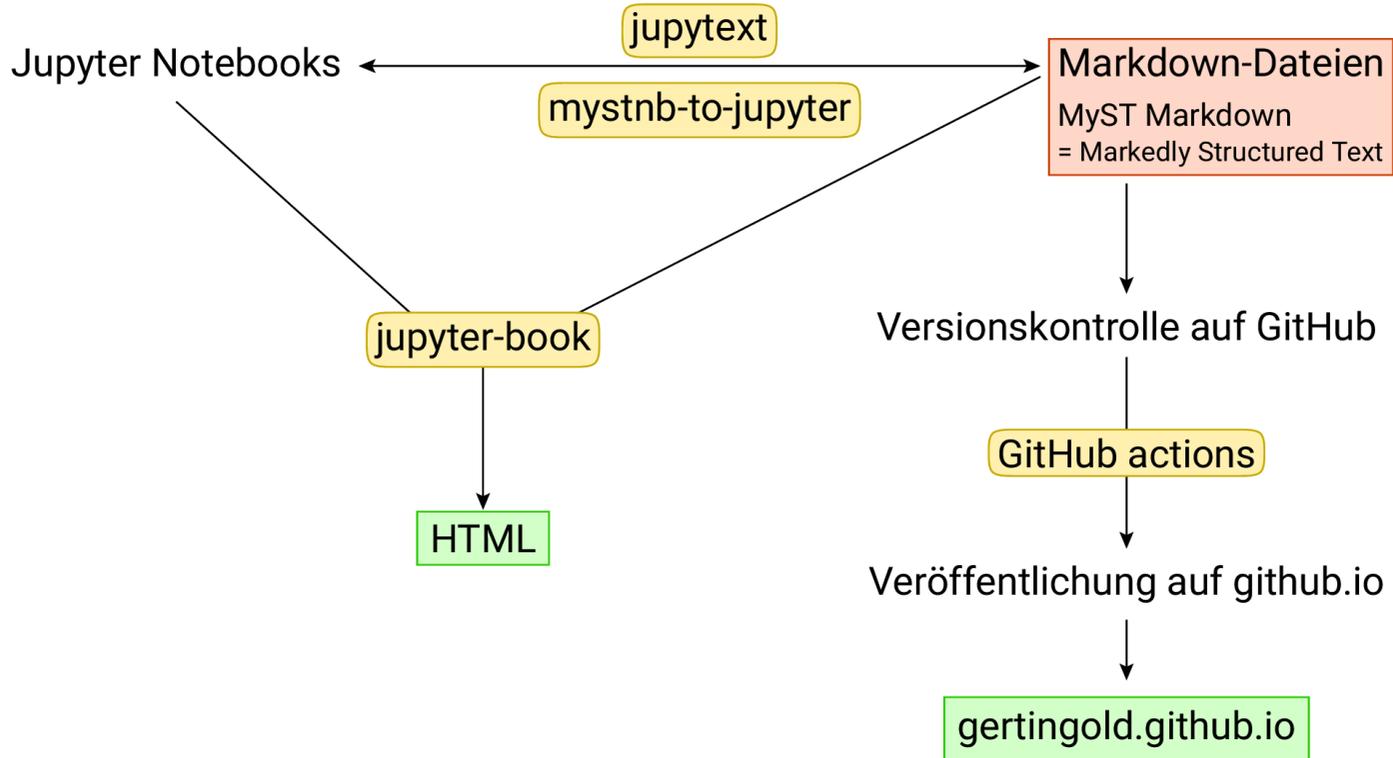
- Integration von Code, auch mit interaktiver Kontrolle, sowie Text, Formeln und Multimediaelementen
- Anbindung an Python-, R-, Julia-Kernel und viele mehr
- Einsatz in Lehre, Datenanalyse und vielem mehr
- Möchte man in der Lehre mit vorgefertigtem Code arbeiten oder eher Code entwickeln?
- Markdown, aber MyST Markdown nur mit einer Extension

Speicherung des Jupyter-Notebooks im JSON-Format

```
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "# Harmonischer Oszillator"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "## Zeitunabhängige Schrödingergleichung\n",
        "\n",
        "$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x)+\frac{m}{2}\omega^2x^2\psi(x) = E\psi(x)$$"
      ]
    }
  ]
}
```

- Versionskontrolle kann Schwierigkeiten bereiten
- Verwendung von `nbstripout` -Filter

MyST Markdown-Dateien als Basis



CommonMark

commonmark.org

Variante 1	Variante 2	Ausgabe
<code># Überschrift 1</code>	<code>Überschrift 1</code> =====	Überschrift 1
<code>## Überschrift 2</code>	<code>Überschrift 2</code> =====	Überschrift 2
<code>##### Überschrift 5</code>		Überschrift 5
<code>*kursiv*</code>	<code>_kursiv_</code>	<i>kursiv</i>
<code>**fett**</code>	<code>__fett__</code>	fett
<code>`inline`</code>		<code>inline</code>

CommonMark

Variante 1	Variante 2	Ausgabe
<pre>* Eintrag 1 * Eintrag 2</pre>	<pre>- Eintrag 1 - Eintrag 2</pre>	<ul style="list-style-type: none">▪ Eintrag 1▪ Eintrag 2
<pre>1. Eintrag 1 2. Eintrag 2</pre>	<pre>1) Eintrag 1 2) Eintrag 2</pre>	<ol style="list-style-type: none">1. Eintrag 12. Eintrag 2
<pre>---</pre>	<pre>***</pre>	<hr/>
<pre>> Blockzitat</pre>		<pre>Blockzitat</pre>

CommonMark

Variante 1

```
[Link](http://a.org)
```

Variante 2

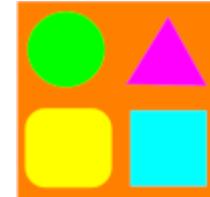
```
[Link][1]  
:  
[1]: http://a.org
```

Ausgabe

Link

```
![Bild](images/bild.png)
```

```
![Bild][1]  
:  
[1]: images/bild.png
```



```
```\n a = 3\n print(a**2)\n```
```

```
````a = 3\n````print(a**2)
```

a = 3
print(a**2)

MyST Markdown anhand von Beispielen

MyST Markdown erweitert CommonMark jupyterbook.org/en/stable/content/

```
es kann auch Fälle geben, in denen man selbst programmieren muss, zum Beispiel wenn man eine Exponentialfunktion auf einen Waschbären anwenden möchte ({numref}`fig:exponential_raccoon`).
```

```
```{figure} images/einleitung/exponential_raccoon.png
```

```

```

```
width: 10cm
```

```
name: fig:exponential_raccoon
```

```

```

```
Originalbild eines Waschbärs (links) und nach Anwendung einer Exponentialfunktion (rechts).
```

```
```
```

Bildbearbeitungsaufgaben sind geeignete Programme verfügbar, aber es kann auch Fälle geben, in denen man selbst programmieren muss, zum Beispiel wenn man eine Exponentialfunktion auf einen Waschbären anwenden möchte ([Abb. 1.1](#)).



Abb. 1.1 Originalbild eines Waschbärs (links) und nach Anwendung einer Exponentialfunktion (rechts).

MyST Markdown anhand von Beispielen

```
````{margin}
````{admonition} Tipp
:class: tip
Eine größere Darstellung von Abbildung können Sie durch Klicken auf das entsprechende Bild erhalten.
````
````
```

Das Beispiel der Digitalkamera schlägt die Brücke vom Alltag ins Labor. Im Grunde genommen ist eine Digitalkamera nichts anderes als ein Messgerät, das



Abb. 1.1 Originalbild eines Waschbärs (links) und nach Anwendung einer Exponentialfunktion (rechts).

Das Beispiel der Digitalkamera schlägt die Brücke vom Alltag ins Labor. Im Grunde genommen ist eine Digitalkamera nichts anderes als ein Messgerät, das optische Information in digitale Daten umwandelt und zur Verfügung stellt, sei es als Rohdaten oder in einer vorverarbeiteten Form. Im Labor wird es erforderlich sein, mit dem Messgerät zu kommunizieren, um Messdaten herunterzuladen. Häufig geschieht dies mit einem vom Hersteller zur Verfügung gestellten Programm, aber man kann sich auch vorstellen, dass man die Schnittstelle des Messgeräts, über die die Messdaten zur Verfügung gestellt werden, mit eigenem Programm anspricht.

Tipp

Eine größere Darstellung von Abbildung können Sie durch Klicken auf das entsprechende Bild erhalten.

MyST Markdown anhand von Beispielen

```
```{admonition} Frage
Was ergibt -2*4+3**2? Was ergibt 6**4//2?
```
```

Frage

Was ergibt `-2*4+3**2`? Was ergibt `6**4//2`?

```
```{admonition} Hinweis
:class: tip
Seit Python 3.9 lässt sich die kleinste darstellbare
Zahl, also 5e-324 mit Hilfe von math.ulp(0)
erhalten.
```
```

Hinweis

Seit Python 3.9 lässt sich die kleinste darstellbare Zahl, also `5e-324` mit Hilfe von `math.ulp(0)` erhalten.

```
```{admonition} Weiterführendes (rechts aufklappen)
:class: toggle
Nach der Kompilierung des obigen C-Programms entsteht al
ein so genanntes Assembler-Programm, das schon sehr masc
von einem Assembler in den von einem Computer les- und a
Maschinencode umgewandelt wird.
```{code-block}
        .file    "bsp_exp.c"
        .text

        :
```

Weiterführendes (rechts aufklappen) Click to show >

MyST Markdown anhand von Beispielen

- Python besitzt eine umfangreiche Standardbibliothek (»Python comes with batteries included«) und das wissenschaftliche Rechnen wird durch eine Vielzahl freier Programmbibliotheken, wie `{program}`NumPy/SciPy``, das wir im Kapitel `{ref}`scipy`` besprechen werden, unterstützt.
- Python hat sich in den letzten Jahren zu einer sehr populären Sprache entwickelt, unter anderem im Bereich der wissenschaftlichen Datenanalyse.

Bei den beiden physikalischen Beobachtungen, die in den letzten Jahren die Aufmerksamkeit einer breiten Öffentlichkeit erregten, nämlich die Beobachtung von Gravitationswellen [`^prd93`] und die Aufnahme des Abbilds eines schwarzen
:

[`^prd93`]: B. P. Abbott et al. , [`Phys. Rev. D 93 , 122003 (2016)`](<https://doi.org/10.1103/PhysRevD.93.122003>). Das Analysepaket PyCBC basiert auf Python und auch das Analysepaket GstLAL enthält einige Pythonskripte.

- Python besitzt eine umfangreiche Standardbibliothek (»Python comes with batteries included«) und das wissenschaftliche Rechnen wird durch eine Vielzahl freier Programmbibliotheken, wie **NumPy/SciPy**, das wir im Kapitel [Numerische Programmbibliotheken am Beispiel von NumPy/SciPy](#) besprechen werden, unterstützt.
- Python hat sich in den letzten Jahren zu einer sehr populären Sprache entwickelt, unter anderem im Bereich der wissenschaftlichen Datenanalyse.

Bei den beiden physikalischen Beobachtungen, die in den letzten Jahren die Aufmerksamkeit einer breiten Öffentlichkeit erregten, nämlich die Beobachtung von Gravitationswellen [\[3\]](#) und die Aufnahme des Abbilds eines schwarzen Loches ([Abb. 1.2](#)) [\[4\]](#), spielte Python bei der

MyST Markdown anhand von Beispielen

Im ersten Beispiel soll der Zusammenhang zwischen der Winkelgeschwindigkeit ω und der Beschleunigung a bei einer Rotationsbewegung untersucht werden. Gemäß der Mechanik erfährt ein Objekt im Abstand r von der Drehachse die Beschleunigung

$$a = r\omega^2$$

Zur experimentellen Untersuchung wird ein Smartphone wie in

Im ersten Beispiel soll der Zusammenhang zwischen der Winkelgeschwindigkeit ω und der Beschleunigung a bei einer Rotationsbewegung untersucht werden. Gemäß der Mechanik erfährt ein Objekt im Abstand r von der Drehachse die Beschleunigung

$$a = r\omega^2$$

Zur experimentellen Untersuchung wird ein Smartphone wie in [Abb. 2.1](#) in einer Salatschleuder montiert, und diese in eine Drehung versetzt. Die im Smartphone vorhandenen Sensoren

⚠️ Warnung

Beim Experimentieren mit dem Smartphone ist Vorsicht geboten, um mechanische Beschädigungen auszuschließen. Das Resultat des

MyST Markdown anhand von Beispielen

```
```{code-block} python

linenos: true
emphasize-lines: 18,21-25,29

import random

def get_result(n_self, n_other):
 result = (n_self - n_other) % 3
 if result == 0:
 return 'Das Spiel endete unentschieden.'
 elif result == 1:
 return 'Du hast gewonnen.'
 return 'Du hast leider verloren.'

objekte = ['Stein', 'Papier', 'Schere']
info_text = ('\n[0] Stein\n'
 '[1] Papier\n'
 '[2] Schere\n\n'
 'Gib eine Zahl zwischen 0 und 2 ein oder -1

while True:
 n_benutzer = int(input(info_text))
 :
```

```
1 import random
2
3 def get_result(n_self, n_other):
4 result = (n_self - n_other) % 3
5 if result == 0:
6 return 'Das Spiel endete unentschieden.'
7 elif result == 1:
8 return 'Du hast gewonnen.'
9 return 'Du hast leider verloren.'
10
11 objekte = ['Stein', 'Papier', 'Schere']
12 info_text = ('\n[0] Stein\n'
13 '[1] Papier\n'
14 '[2] Schere\n\n'
15 'Gib eine Zahl zwischen 0 und 2 ein oder -1 zum Beenden: ')
16
17 while True:
18 n_benutzer = int(input(info_text))
19 if 0 <= n_benutzer <= 2:
20 n_computer = random.randrange(3)
21 print('-'*40)
22 print(f'Benutzer: {objekte[n_benutzer]}')
23 print(f'Computer: {objekte[n_computer]}')
24 print(get_result(n_benutzer, n_computer))
25 print('-'*40)
26 elif n_benutzer == -1:
27 break
28 else:
29 print('\n*** ungültige Eingabe\n')
```

# MyST Markdown anhand von Beispielen

Möchte man aus den Werten zweier Variablen eine komplexe Zahl konstruieren, geht dies mit der zweiten der gerade genannten Methoden sehr einfach

```
```{code-cell} python
x = 1
y = 2
z1 = complex(x, y)
z2 = complex(x, -y)
z1/z2
```
```

Falls man die Funktion `{func}`complex`` nicht verwenden möchte, muss man beachten, dass die folgenden beiden Wege

Möchte man aus den Werten zweier Variablen eine komplexe Zahl konstruieren, geht dies mit der zweiten der gerade genannten Methoden sehr einfach

```
x = 1
y = 2
z1 = complex(x, y)
z2 = complex(x, -y)
z1/z2
```

```
(-0.6+0.8j)
```

Falls man die Funktion `complex()` nicht verwenden möchte, muss man beachten, dass die

# MyST Markdown anhand von Beispielen

```
```{code-cell} python
---
tags: [raises-exception]
---
x = 18
y = 9
z = x+yj
```
```

```
x = 18
y = 9
z = x+yj
```

```

NameError Traceback (most recent call last)
Cell In[62], line 3
 1 x = 18
 2 y = 9
----> 3 z = x+yj

NameError: name 'yj' is not defined
```

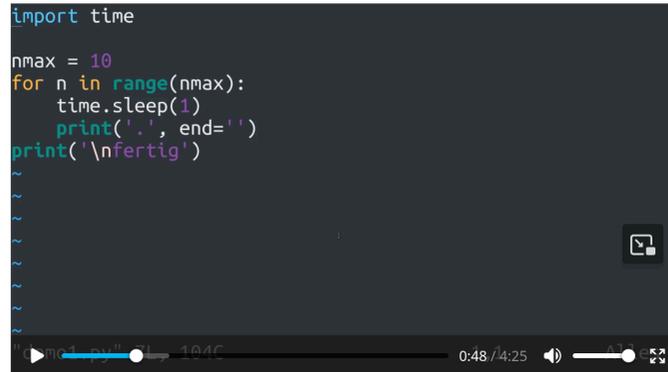
# MyST Markdown anhand von Beispielen

auch im nächsten Abschnitt noch einmal eine Rolle spielen, wenn wir die Ausgabe von Daten in eine Datei besprechen.

```
<video width="640" height="360" controls>
 <source src="https://gertingold.github.io/resources/flush.webm" type="video/webm">
Ihr Browser unterstützt nicht das video-Tag.
</video>
```

```
(readfile)=
Lesen von Dateien
```

auch im nächsten Abschnitt noch einmal eine Rolle spielen, wenn wir die Ausgabe von Daten in eine Datei besprechen.

A screenshot of a video player showing Python code. The code is as follows:

```
import time
nmax = 10
for n in range(nmax):
 time.sleep(1)
 print('.', end='')
print('\nfertig')
```

The video player interface includes a progress bar at the bottom showing 0:48 / 4:25, a volume icon, and a full-screen icon.

## 7.2. Lesen von Dateien

## \_conf.yml

```
title : Einführung in Prinzipien der Programmierung
author : Gert-Ludwig Ingold
copyright : "CC-BY"
logo : ""

only_build_toc_files: true

repository:
 url: https://github.com/gertingold/epriprog
 path_to_book: epriprog

html:
 use_repository_button: false
 use_issues_button: false
 use_edit_page_button: false
 home_page_in_navbar: false

latex:
 latex_documents:
 targetname: epriprog.tex
 latex_elements:
 papersize: a4paper
```

## \_toc.yml

```
root: intro
options:
 numbered: true
chapters:
 - file: einleitung
 - file: vorschau
 - file: datentypen
 - file: kontrollstrukturen
 - file: funktionen
 - file: sequenzen
 - file: einausgabe
 - file: scipy
 - file: objektorientiert
 - file: zahlensysteme
 - file: floats
 - file: unicode
```

1. Einleitung
2. Eine Vorschau
- 3. Einfache Datentypen, Variablen und Zuweisungen**
4. Kontrollstrukturen
5. Funktionen
6. Zusammengesetzte Datentypen
7. Ein- und Ausgabe
8. Numerische Programmbibliotheken am Beispiel von NumPy/SciPy
9. Objektorientiertes Programmieren
10. Anhang: Zahlensysteme
11. Anhang: 64-Bit-Gleitkommazahlen nach IEEE-Standard 754
12. Anhang: Unicode

# 3. Einfache Datentypen, Variablen und Zuweisungen

In [Kapitel 2](#) hatten wir bereits verschiedene Datentypen kennengelernt. Hierzu gehören ganze Zahlen, die zum einen durch die explizite Umwandlung der Benutzereingabe in eine ganze Zahl oder als Ergebnis der `randrange()`-Funktion erzeugt wurden. Des Weiteren kamen Wahrheitswerte vor. Ganz explizit war dies bei `True` der Fall, das zur Konstruktion einer Dauerschleife verwendet wurde.

Diese beiden Datentypen, ganze Zahlen und Wahrheitswerte gehören zu den einfachen Datentypen. Daneben gibt es auch zusammengesetzte Datentypen, für die wir in [Kapitel 2](#) ebenfalls schon Beispiele gesehen hatten. Dazu gehören die Zeichenketten, zum Beispiel in den Zeilen 12 bis 15 des Beispielprogramms, aber auch die durch eckige Klammern gekennzeichnete Liste in Zeile 11.

In diesem Kapitel werden wir uns zunächst den einfachen Datentypen zuwenden und die zusammengesetzten Datentypen in einem späteren Kapitel besprechen. Dabei werden wir vor allem neben den ganzen Zahlen noch weitere numerische Datentypen kennenlernen, die für natur- und ingenieurwissenschaftliche Anwendungen große Bedeutung besitzen.

## 3.1. Ganze Zahlen

Wir beginnen bei der Besprechung der numerischen Datentypen mit den ganzen

</> .ipynb

.md

.pdf

1. Ganze Zahlen
2. Gleitkommazahlen
3. Funktionen für reelle Zahlen
- 3.4. Komplexe Zahlen
- 3.5. Variablen und Zuweisungen
- 3.6. Wahrheitswerte
- 3.7. Formatierung von Ausgaben

```
program datentyp
 integer :: n
 real :: x

 n = 2
 x = n
 write(*, *) n, x
end program datentyp
```

erzeugt die Ausgabe

```
2 2.00000000
```

Gemäß der Deklaration zu Beginn des Programms handelt es sich bei der Variable `n` um einen Integer, während `x` ein Float ist, der in Fortran mit `real` bezeichnet wird. In der hervorgehobenen Zeile findet bei der Zuweisung automatisch

einfach abgeschnitten wird:

```
int(2.7)
```

```
2
```

Bereits das Anhängen eines Punktes genügt, damit Python die Zahl als Float interpretiert:

```
print(type(2.))
```

```
<class 'float'>
```

Für Floats gibt es zwei mögliche Schreibweisen. Zum einen die kann man die Dezimalbruchschriftweise verwenden,

# GitHub

github.org

- Plattform für Software-Versionskontrolle und Projektzusammenarbeit
- freier persönlicher Account

## github.io

- Webseiten unter *username.github.io* unter Verwendung eines öffentlichen GitHub-Repositories mit diesem Namen
- Erzeugung des Inhalts aus anderen GitHub-Repositories mit Hilfe von GitHub-Actions in einem Zweig `gh-pages`
- statische Webseiten, also z.B. keine PHP-Skripten
- Beschränkung der Seitengröße und Bandbreite, nicht für kommerziellen Gebrauch

# GitHub Actions

## `.github/workflows/deploy.yml`

```
uses: actions/setup-python@v5
with:
 python-version: '3.11'

- name: install dependencies
 run: |
 conda install -y python=3.11
 conda env update --file environment.yml --name base
 echo $CONDA/bin >> $GITHUB_PATH

- name: build the book
 run: |
 jupyter-book build epriprogram

- name: deploy book's HTML to gh-pages branch
 uses: peaceiris/actions-gh-pages@v4
 with:
 github_token: ${{ secrets.GITHUB_TOKEN }}
 publish_dir: epriprogram/_build/html
```

## `environment.yml`

```
name: epriprogram
channels:
 - default
 - conda-forge
dependencies:
 - python==3.11
 - scipy
 - matplotlib
 - jupyter-book
```

# mystmd

[mystmd.org](https://mystmd.org)

- neben Jupyter Book zweites (und jüngerer) Hauptprojekt von [executable{books}](#)
- Javascript statt Python, schneller und mehr Funktionalität
- Überlapp mit Jupyter Book
  - Jupyter Book Quellen verwendbar
  - Intersphinx für Verlinkung mit Sphinx-Dokumentationen
- vor allem [Templates](#) für Journale (derzeit 2 `docx`, 18 `tex`, 2 `typst`) sowie 2 Templates für Webseiten (article und book)
- Möglichkeit für interaktive Inhalte
- Lokalisierung??

# Ein paar Eindrücke von `mystmd`

Dies ist ein MyST-Beispiel für den LinuxInfoTag 2024, in dem es allgemein um `<wiki:Linux>` geht.

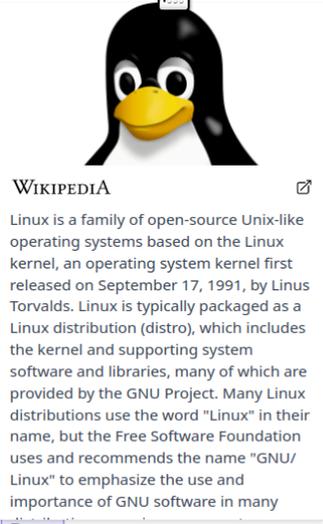
Beim LinuxInfoTag 2017 habe ich ein `<img alt="Tux the penguin" data-bbox="358 208 448 362"/>` gezeigt. Man kann verweisen: `<img alt="Tux the penguin" data-bbox="358 208 448 362"/>`, aber in der Bildüberschrift steht trotzdem

Im Hauptteil steht eine berühmte `<math>E=mc^2</math>`, die die Gleichung

Hier wurde die erste Beobachtung von Gravitationswellen publiziert

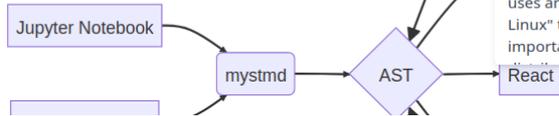
Intersphinx-Link zu einer Seite der Jupyter Book Dokumentation

Dieses Diagramm ist mit mermaid erstellt und stammt aus dem



WIKIPEDIA

Linux is a family of open-source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged as a Linux distribution (distro), which includes the kernel and supporting system software and libraries, many of which are provided by the GNU Project. Many Linux distributions use the word "Linux" in their name, but the Free Software Foundation uses and recommends the name "GNU/Linux" to emphasize the use and importance of GNU software in many



Dies ist ein MyST-Beispiel für den LinuxInfoTag 2024, in dem es allgemein um `<wiki:Linux>` geht.

Beispielcode, siehe  
[github.com/gertingold/lit2024/tree/main/myst-example](https://github.com/gertingold/lit2024/tree/main/myst-example)

# Ein paar Eindrücke von `mystmd`

Beim LinuxInfoTag 2017 habe ich ein `Bild` gezeigt. Man kann auf das Bild auch mit seiner Nummer verweisen: `Abb. 1`, aber in der Bildüberschrift steht trotzdem "Figure 1".



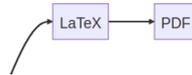
Figure 1: Ein exponentierter Waschbär

die Gleichungsnummer (1) besitzt.

onswellen publiziert: [Abbott et al. \(2016\)](#)

Dokumentation: [GitHub Pages and Actions](#)

tammt aus der MYS-T-Dokumentation:



Beim LinuxInfoTag 2017 habe ich ein `[Bild](#exponential_raccoon)` gezeigt. Man kann auf das Bild auch mit seiner Nummer verweisen: `[Abb. %s](#exponential_raccoon)`, aber in der Bildüberschrift steht trotzdem "Figure 1".

verweisen: `Abb. 1`, &



Figure 1: Ein exponentierter Waschbär

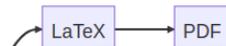
# Ein paar Eindrücke von `mystmd`

Im Hauptteil steht eine berühmte `Gleichung`, die die Gleichungsnummer (1) besitzt.

Hier v  $E = mc^2$  (1) `vott et al. (2016)`

Intersphinx-Link zu einer Seite der Jupyter Book Dokumentation: `GitHub Pages and Actions`

Dieses Diagramm ist mit mermaid erstellt und stammt aus der MyST-Dokumentation:



## Hauptteil



Figure 1: Ein exponentierter Waschbär

Dies ist eine Gleichung:

$$E = mc^2 \tag{1}$$

## 01-einleitung.md

Im Hauptteil steht eine berühmte `[Gleichung](#einstein)`, die die Gleichungsnummer `{numref}`einstein`` besitzt.

## 02-hauptteil.md

Dies ist eine Gleichung:

```
```{math}
:label: einstein
E = mc^2
```
```

# Ein paar Eindrücke von `mystmd`

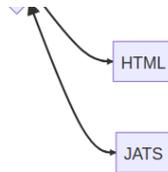
n Gravitationswellen publiziert: [Abbott et al. \(2016\)](#)

MySTer Bo  
:tellt un

Abbott, B. P., Abbott, R., Abbott, T. D., Abernathy, M. R., Acernese, F., Ackley, K., Adams, C., Adams, T., Addesso, P., Adhikari, R. X., Adya, V. B., Affeldt, C., Agathos, M., Agatsuma, K., Aggarwal, N., Aguiar, O. D., Aiello, L., Ain, A., Ajith, P., ... Zweizig, J. (2016). GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. *Physical Review D*, 93(12). [10.1103/physrevd.93.122003](https://doi.org/10.1103/physrevd.93.122003)

Hier wurde die erste Beobachtung von Gravitationswellen publiziert: [@doi:10.1103/PhysRevD.93.122003](https://doi.org/10.1103/PhysRevD.93.122003)

MyST Markdown



- Am Ende wird automatisch eine Referenzenliste erzeugt.

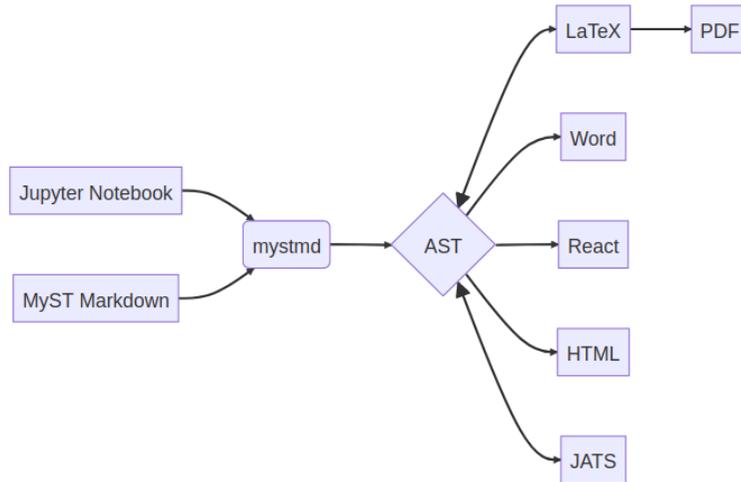
## References

1. Abbott, B. P., Abbott, R., Abbott, T. D., Abernathy, M. R., Acernese, F., Ackley, K., Adams, C., Adams, T., Addesso, P., Adhikari, R. X., Adya, V. B., Affeldt, C., Agathos, M., Agatsuma, K., Aggarwal, N., Aguiar, O. D., Aiello, L., Ain, A., Ajith, P., ... Zweizig, J. (2016). GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. *Physical Review D*, 93(12). [10.1103/physrevd.93.122003](https://doi.org/10.1103/physrevd.93.122003)

# Ein paar Eindrücke von `mystmd`

Intersphinx-Link zu einer Seite der Jupyter Book Dokumentation: [GitHub Pages and Actions](#)

Dieses Diagramm ist mit mermaid erstellt und stammt aus der MyST-Dokumentation:



Intersphinx-Link zu einer Seite der Jupyter Book Dokumentation: `[](myst:jupyterbook#publish/gh-pages)`

Dieses Diagramm ist mit mermaid erstellt und stammt aus der MyST-Dokumentation:

```
```{mermaid}
flowchart LR
  A[Jupyter Notebook] --> C
  B[MyST Markdown] --> C
  C(mystmd) --> D{AST}
  D <--> E[LaTeX]
  E --> F[PDF]
  D --> G[Word]
  D --> H[React]
  D --> I[HTML]
  D <--> J[JATS]
```
```

# Nützliche Links

- Executable Book Project: [executablebooks.org](https://executablebooks.org)
- Jupyter Book: [jupyterbook.org](https://jupyterbook.org)
- MyST: [mystmd.org](https://mystmd.org)
  
- Quellen zu diesem Vortrag: [github.com/gertingold/lit2024](https://github.com/gertingold/lit2024)
  
- Beispiel mit Jupyter Book: [github.com/gertingold/epriprogram](https://github.com/gertingold/epriprogram)
- Beispiel mit Sphinx: [github.com/gertingold/tools4scicomp](https://github.com/gertingold/tools4scicomp)

Fragen?